VLSI DESIGN
1996, Vol. 4, No. 1, pp. 33-40
Reprints available directly from the publisher
Photocopying permitted by license only

Time- and Cost-Optimal Parallel Algorithms for the Dominance and Visibility Graphs*

D. BHAGAVATHI, H. GURLA, S. OLARIU† and J. L. SCHWING
Department of Computer Science, Old Dominion University, Norfolk, VA 23529-0162

J. ZHANG
Department of Math and Computer Science, Elizabeth City State University, Elizabeth City, NC 27909

(Received December 17, 1992, Revised July 30, 1993)

The compaction step of integrated circuit design motivates associating several kinds of graphs with a collection of non-overlapping rectangles in the plane. These graphs are intended to capture various visibility relations amongst the rectangles in the collection. The contribution of this paper is to propose time- and cost-optimal algorithms to construct two such graphs, namely, the dominance graph (DG, for short) and the visibility graph (VG, for short). Specifically, we show that with a collection of n non-overlapping rectangles as input, both these structures can be constructed in $\theta(\log n)$ time using n processors in the CREW model.

Keywords: *CAD, Compaction, Constraint Graph, Dominance Graph, Visibility Graph, Parallel Algorithms, Lower Bounds, Time-Optimal Algorithms, CREW*

1 INTRODUCTION

Two important design methodologies central to the fabrication of integrated circuits are *symbolic layout* and *compaction*. The symbolic layout involves representing a mask layout using symbols and languages to manipulate these symbols; compaction changes the geometry of the topological design to produce a smaller layout while enforcing the design rules and constraints [10, 13]. Masks used in the process of integrated circuit layout design are usually represented by a set of rectangles in two dimensions, with edges parallel to the axes (henceforth referred to as *iso-oriented*) [10, 11].

As it turns out, most of the steps in the compaction process are simplified by using a symbolic description of the layout. It is quite natural, therefore, to express the compaction problem as a problem involving a collection of iso-oriented, non-overlapping, rectangles in the plane. For simplicity reasons the compaction process is one-dimensional, i.e. the components are moved in the x-direction or y-direction only, or two-dimensional where in a single step selected components are moved in both directions [12, 14]. Typically, the compaction process is based on the notion of *constraint graph* (CG, for short). Each vertex of the constraint graph corresponds to a rectangle or to a group of electrically equivalent elements; the edges of the constraint graph correspond to the design constraints that must be satisfied by the circuit [8, 12, 14].

However, in general the constraint graph is rather complicated, with many extraneous edges [14]. From a computational standpoint [5] one is motivated to investigate properties of simpler graphs that provide useful heuristic solutions to a good number of special cases of compaction. An important such graph is the *visibility graph* (VG, for short) studied by several workers [5, 6, 12]. Consider a collection of iso-oriented, non-overlapping, rectangles in the plane. Two rectangles R and R' are said to be *visible* if there exists a horizontal line intersecting R and R' and not intersecting any other rectangle that lies between R and R'. The visibility graph has the set of rectangles as its vertices, with two vertices

*Work supported in part by NASA under grant NCC1-99 and by the NSF under grant CCR-9407180; © 1994 IEEE. Reprinted with permission from *Proceedings of VLSI Design'94*, Calcutta, India, January 1994

† Address for Correspondence: Prof. Stephan Olariu, Department of Computer Science, Old Dominion University, Norfolk, VA 23529-0162, email: olariu@cs.odu.edu, Phone: (804) 683-4417, FAX: (804) 683-4900

joined by an edge if and only if the corresponding rectangles are visible.

In this work we shall also define a subgraph of the visibility graph that we refer to as the *dominance graph* (DG, for short). Consider, again, a collection of iso-oriented, non-overlapping, rectangles in the plane. We say that a rectangle A is *above* rectangle B if some points in A and B share the same x-coordinate with the point in A having a larger y-coordinate. A rectangle A is *directly above* B if A is above B and no rectangle C is such that A is above C, and C is above B. The dominance graph of a collection of non-overlapping rectangles is a directed graph whose vertices are precisely the rectangles in the collection; two vertices u and v are linked by a directed edge (u, v) when the rectangle corresponding to u is directly above the rectangle corresponding to v. It is easy to see that the notion of direct aboveness captures the intuitive idea that for compaction purposes we only need consider rectangles that are "neighbors" in the sense that no other rectangles lie between them. It is not hard to see that the dominance, visibility, and constraint graphs are related by DG $\subset$ VG $\subset$ CG.

In this paper we assume the Parallel Random Access Machine model which consists of synchronous processors, each having access to a common memory. At each step, every processor performs the same instruction, with a number of processors masked out. In a Concurrent Read Exclusive Write PRAM model (CREW, for short), several processors may simultaneously read the same memory location, but exclusive access is used for writing. The interested reader is referred to [7] for a more detailed discussion on the CREW model. The *cost* of a parallel algorithm [7] is taken to be the product of its running time and the number of processors used. If the cost of a parallel algorithm matches the sequential lower bound for the given problem, the parallel algorithm is termed *cost-optimal*. A parallel algorithm is termed *time-optimal* within a given computational model is no other parallel algorithm solving the same problem runs faster in that model.

The main contribution of this paper is to provide time- and cost-optimal algorithms for the problems of constructing the dominance and visibility graphs of a collection of n iso-oriented, non-overlapping, rectangles in the plane. Our first result is to show that the problem of constructing the dominance graph has a lower bound of $\Omega(n \log n)$ by reducing the ELEMENT UNIQUENESS [4] problem to it. Next, we derive time-lower bounds for the problems of computing the dominance and visibility graphs in the CREW model. Specifically, we show that any parallel algorithm solving these problems must take $\Omega(\log n)$ time in the CREW even if an infinite number of processors and memory cells are used. Finally, we propose time- and cost-optimal algorithms to construct the dominance and visibility graphs running in $\theta(\log n)$ time and using $O(n)$ processors in the CREW model of computation.

The remainder of this work is organized as follows: Section 2 presents our lower bounds; Section 3 presents the details of our time-optimal algorithm to construct the dominance graph; Section 4 discusses the details of the proposed time-optimal algorithm for the visibility graph; finally, Section 5 summarizes the results and poses some open problems.

2 LOWER BOUNDS

The purpose of this section is to provide lower bounds that establish both the time- and cost-optimality of our algorithms in the CREW model of computation.

We first show that any algorithm that correctly constructs the domination graph of a collection of n non-overlapping rectangles in the plane must perform at least $\Omega(n \log n)$ operations. Our arguments rely on a well-known lower bound for the following classic problem.

ELEMENT UNIQUENESS: Given n real numbers x_1, x_2, ..., x_n, decide whether any two of them are equal.

Proposition 2.1. [4] ELEMENT UNIQUENESS has a lower bound of Ω ($n \log n$) in the algebraic tree model of computation. □

Lemma 2.2. The problem of constructing the domination graph of a collection of n non-overlapping rectangles in the plane has a lower bound of $\Omega(n \log n)$ in the algebraic decision-tree model.

Proof. Assume that the input to the ELEMENT UNIQUENESS problem is a set x_1, x_2, ..., x_n of real numbers. Construct a family R_1, R_2, ..., R_n of n *degenerate* rectangles in the plane by letting every R_i be specified by its lower-left and upper-right corners both defined as (x_i, i). In other words, every rectangle R_1 reduces to the point (x_i, i) in the plane. Notice that by construction the rectangles are non-overlapping. Now consider the dominance graph corresponding to this family of rectangles: by scanning the degrees of vertices of this graph we find out in $O(n)$ time whether any two of the real numbers x_1, x_2, ..., x_n are equal. The conclusion follows from Proposition 2.1. □

Next, we note that Schlag *et al.* [12] have established a similar lower bound for the problem of constructing the visibility graph of n rectangles in the plane. Specifically, a variant of the following result was established in [12].

Proposition 2.3. The problem of constructing the visibility graph of a collection of n non-overlapping rectangles in the plane has a lower bound of $\Omega(n \log n)$ in the algebraic decision-tree model.

In fact, Proposition 2.3 can also be proved directly in a way essentially similar to the proof of Lemma 2.2.

Our next goal is to establish a time lower bound for the tasks of constructing the dominance and visibility graphs in the CREW model. For this purpose, we rely on a fundamental result of Cook *et al.* [3]. To make this paper self-contained, we define the problem and state the relevant result from [3].

OR: Given n bits $b_1, b_2, \ldots, b_n$, compute their logical OR.

Proposition 2.4. [3] OR has a time lower bound of $\Omega(\log n)$ on the CREW, regardless of the number of processors and memory cells used.

Theorem 2.5. The problem of constructing the dominance graph of a set of n non-overlapping rectangles in the plane has a lower bound of $\Omega(\log n)$ on the CREW, regardless of the number of processors and memory cells used.

Proof. To establish the lower bound, we only need show that the solution to the dominance graph problem for n iso-oriented, non-overlapping, rectangles in the plane yields a solution to the OR problem. Let $b_1, b_2, \ldots, b_n$ be the input to the OR problem. We construct a collection $R_0, R_1, \ldots, R_n$ of $n + 1$ non-overlapping rectangles. We specify each rectangle by an ordered pair consisting of the coordinates of its lower-left and upper-right corners, respectively. More precisely, we set:

- $R_0 = ((1, n + 1), (n + 2, n + 1.5))$;
- for all i $(1 \leq i \leq n)$,
  $R_i = ((i, n - i + 1), (i + 2, n - i + 1.5))$, whenever $b_i = 0$;
  $R_i = ((i, n - i + 1), (i + 1, n - i + 1.5))$, whenever $b_i = 1$.

(Figure 1 illustrates this construction for the input sequence $b_1 = 0, b_2 = 0, b_3 = 1, b_4 = 1, b_5 = 0$.) Clearly the answer to the OR problem is 0 if and only if the out-degree of the vertex corresponding to R_0 is 1. Therefore, once the dominance graph is available, one can find the answer to OR in O(1) time. By virtue of Proposition 2.4, any algorithm that computes the dominance graph must take $\Omega(\log n)$ time in the worst case. This completes the proof of the theorem. □

Theorem 2.6. The problem of constructing the visibility graph of a set of n iso-oriented, non-overlapping rectangles in the plane has a lower bound of $\Omega(\log n)$ on the CREW, regardless of the number of processors and memory cells used.

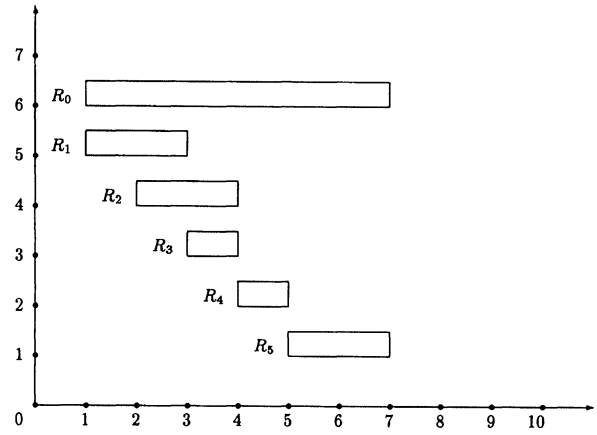

FIGURE 1 Illustrating the lower bound for dominance graph.

Proof. The claim will be established by showing that a solution of the visibility graph problem implies a solution for OR. Let the bit string $b_1, b_2, \ldots b_n$ be the input to the OR problem. Corresponding to this input, we construct a collection of $n + 2$ iso-oriented, non-overlapping, rectangles $R_0, R_1, \ldots, R_n, R_{n+1}$. We specify each rectangle by an ordered pair consisting of the coordinates of its lower-left and upper-right corners, respectively. More precisely, we set:

- $R_0 = ((0, 0), (0.5, 2))$, $R_{n+1} = ((n + 1, 0), (n + 1.5, 2))$ and
- for all i $(1 \leq i \leq n)$
  $R_i = ((i, 0), (i + 0.5, 3))$, whenever $b_i = 1$, and
  $R_i = ((\mathrm{i}, 0), (\mathrm{i} + 0.5, 1))$, whenever $b_i = 0$.

(Figure 2 illustrates this construction for the input sequence $b_1 = 0, b_2 = 0, b_3 = 1, b_4 = 1, b_5 = 0$.) Clearly the solution to the OR problem is 0 if and only if R_0 and R_{n+1} are visible. Thus, once the visibility graph is available, one can find the answer to OR in O(1) time. Therefore, Proposition 2.4 guarantees that any algorithm that constructs the visibility graph must take $\Omega(\log n)$ time in the worst case. □

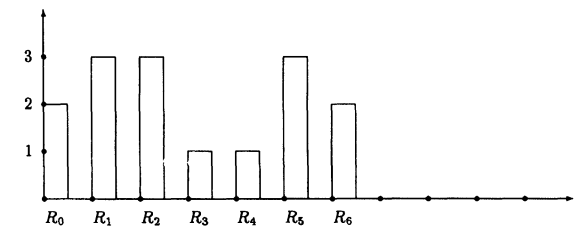

FIGURE 2 Illustrating the lower bound for visibility graph.

3 CONSTRUCTING THE DOMINANCE GRAPH

The purpose of this section is to exhibit a time- and cost-optimal algorithm to construct the dominance graph of a set of n iso-oriented, non-overlapping, rectangles R_1, R_2, ..., R_n in the plane. For convenience, we assume that every rectangle R_i is represented by a set of four edges, $R = \{l_i, r_i, t_i, b_i\}$, with l_i and r_i standing for the left and right vertical edges respectively; similarly, we let t_i and b_i denote the top and bottom horizontal edges, respectively. Every edge is represented by its two endpoints. To avoid tedious and inconsequential housekeeping details, we assume that all rectangles have distinct x and y coordinates.

Draw from each endpoint of a horizontal edge t_i (resp. b_i) ($1 \leq i \leq n$) upward (resp. downward) pointing half-lines. Each of these half-lines terminates when it encounters the horizontal edge of another rectangle, or continues to infinity. It is customary to refer to these half-lines as *trapezoidal* lines. What results is a partition of the plane into rectangular regions (bounded or unbounded). Such a decomposition is termed a *trapezoidal* decomposition (see Figure 3). Furthermore, it is fairly easy to see that the number of trapezoids in the decomposition is $O(n)$.

Note that every bounded trapezoid T has its horizontal edges determined by the horizontal edges of two rectangles in our collection; the vertical edges are either trapezoidal lines or combinations of trapezoidal lines and the vertical edge of some rectangle (refer to Figure 3 for an illustration). As we are about to point out, the trapezoidal decomposition is closely related to the notion of direct aboveness.

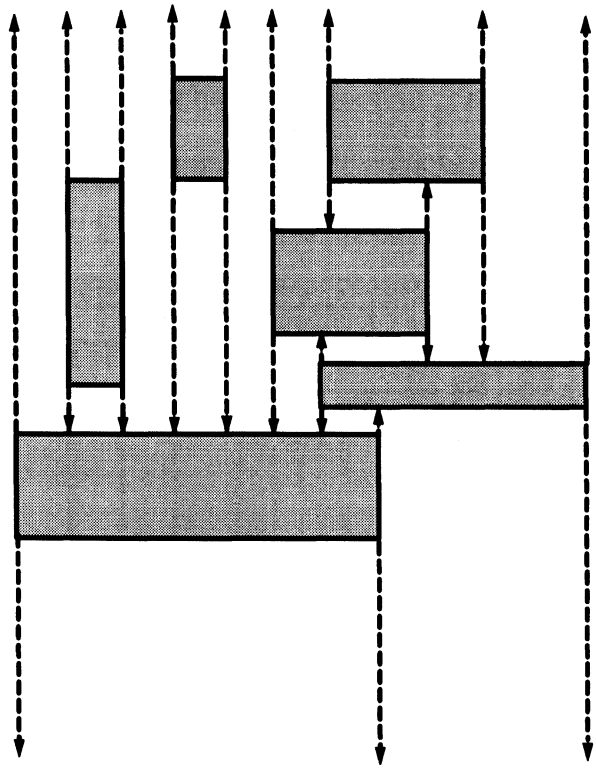

FIGURE 3 A trapezoidal decomposition.

Lemma 3.1. Let R_u and R_v be rectangles such that b_v has a larger y-coordinate than t_u. R_v is directly above R_u if, and only if, there exists a trapezoid determined by b_v, t_u and two trapezoidal lines originating at the endpoints of b_v and t_u.

Proof. To begin, note that if R_v is directly above R_u then there exists a trapezoid satisfying the requirements of the lemma (refer to Figure 4).

Conversely, suppose that the trapezoidal region consisting of b_v, t_u and two trapezoidal lines originating at the endpoints of b_v and t_u exists. It is easy to see that no other rectangle in the collection can be between R_v and R_u, and the conclusion follows. □

Lemma 3.1 motivates the following approach to constructing the dominance graph. Begin by performing the trapezoidal decomposition of the plane and assign one processor to each corner of every rectangle in the collection. Every processor is responsible for detecting a trapezoid satisfying the requirements of Lemma 3.1. The dominance graph DG will be returned by its adjacency lists structure: that is, for every vertex u of DG we return a linked list that contains all the vertices v for which (u, v) is an edge in DG. The details are spelled out as follows.

Algorithm Construct_Dominance_Graph; {Input: a collection R_1, R_2, ..., R_n of n iso-oriented non-overlapping rectangles in the plane;
Output: the corresponding dominance graph DG represented by its adjacency lists;}

Step 1. Sort all the edges l_i ($1 \leq i \leq n$) in increasing order of their x-coordinates;

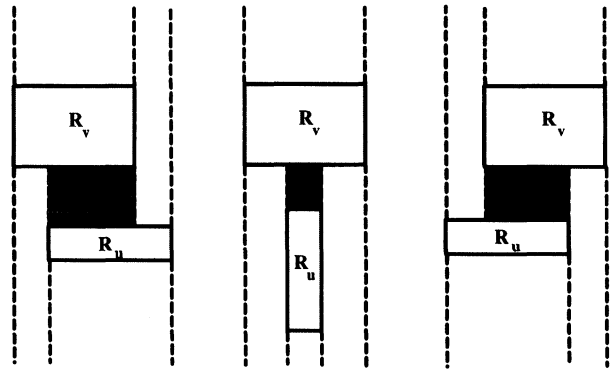

FIGURE 4 The trapezoid T satisfies the conditions of Lemma 3.1.

Step 2. Divide the plane into trapezoids by using the optimal parallel trapezoidal decomposition of [1];

Step 3. Assign one processor to each corner of every rectangle in the collection; for ($1 \leq i \leq n$) let P_1^i, P_2^i, P_3^i, P_4^i be the processors assigned to R_I, with P_1^i assigned to the upper-left corner and proceeding counterclockwise (refer to Figure 5);

Step 4. For all i, j ($1 \leq i \leq n$; $1 \leq j \leq 4$) in parallel processor P_j^i identifies the following trapezoidal line segments:

- a_j^i starting from the point on which P_j^i is assigned and pointing upwards (downwards);
- c_j^i the nearest vertical line segment to the right of a_j^i and parallel to a_j^i;
- b_j^i and d_j^i the two horizontal line segments which bound a_j^i and c_j^i (see Figure 5 for an illustration);

Step 5. For all i, j ($1 \leq i \leq n$; $1 \leq j \leq 4$) processor P_j^i marks itself *active* if the rectangular region determined by a_j^i, b_j^i, c_j^i, and d_j^i satisfies the conditions of Lemma 3.1; otherwise, P_j^i marks itself *inactive* and will never be active again;

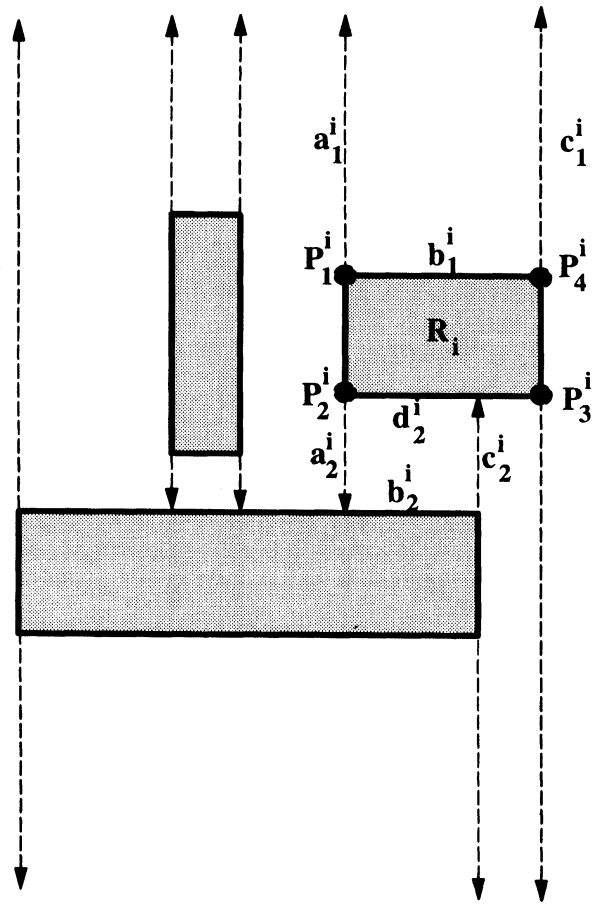

FIGURE 5 Illustrating Steps 3 and 4 of Construct_Dominance_Graph.

Step 6. Every active processor P_j^i records (s, t) in its own memory, whenever rectangle R_s is above rectangle R_t;

Step 7. Sort the ordered pairs (s, t) produced in Step 6 by their first component;

Step 8. In the sorted list mark the first element in every group of ordered pairs having the same first component; what results are the adjacency lists of the dominance graph.

To summarize our findings we state the following result.

Theorem 3.2. The dominance graph corresponding to a collection of n iso-oriented, non-overlapping, rectangles in the plane can be constructed in $O(\log n)$ time using $O(n)$ processors in the CREW. Furthermore, this is both time- and cost-optimal in this model of computation.

Proof. The correctness follows directly from Lemma 3.1. We turn our attention to the complexity. First, we note that since no write conflicts arise, the computation can be performed in the CREW model.

Notice that Step 1 takes $O(\log n)$ time using $O(n)$ processors if we use Cole's sorting algorithm [5]. Next, Step 2 uses the algorithm in [1] running in $O(\log n)$ time and using $O(n)$ processors in the CREW model. Step 3 involves a simple assignment operation, and thus requires O(1) time. Note that in Step 4 every processor P_j^i takes O(1) time to determine the lines a_j^i, b_j^i, c_j^i, and d_j^i. To clarify this last point, note that a_j^i, b_j^i, c_j^i, and d_j^i belong to the same trapezoid in the collection and so the search of Step 4 can be performed in constant time. Furthermore, once this information is known, P_j^i marks itself active only if these lines satisfy the conditions of Lemma 3.1: this is done in O(1) time; Further, Step 7 can be implemented in $O(\log n)$ time with $O(n)$ processors by using Cole's sorting algorithm. Finally, Step 8 can be implemented in O(1) time with $O(n)$ processors in the obvious way. By Lemma 2.2 our algorithm is cost-optimal; by Theorem 2.5 the algorithm is also time-optimal. This completes the proof of Theorem 3.2. □

4 CONSTRUCTING THE VISIBILITY GRAPH

Consider, again, a collection = $\{R_1, R_2, \ldots, R_n\}$ of n iso-oriented, non-overlapping, rectangles in the plane. For the purpose of constructing the visibility graph, it is convenient to abstract rectangles as vertical line segments as described below. In this context, the compaction is referred to as "stick" compaction [5, 14].

Specifically, from the collection we obtain a collection $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$ of n disjoint vertical line segments in the plane, such that segment s_i corresponds to the leftmost vertical edge of rectangle R_i. We say that a pair s_i, s_j of such segments is a *visible pair* (or, simply, that they *see each other* [5]) if there exists a horizontal line intersecting s_i and s_j and not intersecting any other line segment s_k that lies between s_i and s_j (see Figure 6). It is not hard to see that the visibility graph of the original set of rectangles is precisely the visibility graph of the resulting collection of vertical line segments.

For every such line segment s_i ($1 \leq i \leq n$) we let (x_i, t_i) and (x_i, b_i) denote its top and bottom endpoints, respectively. To avoid inconsequential and tedious housekeeping details, we assume that all segments have distinct x and y coordinates. As noted in [5, 12] this can always be done without any loss of generality.

Draw from each endpoint of every segment right and left-pointing half-lines. Each of these half-lines terminates when it encounters another vertical segment in the collection, or continues to infinity. What results is a trapezoidal decomposition of the plane (refer to Figure 7).

It is easy to confirm that every bounded trapezoid T has its horizontal edges determined by trapezoidal lines; the vertical edges correspond to sub-segments of two vertical segments in our collection. As we are about to point out, the trapezoidal decomposition is closely related to the notion of visibility among vertical segments in the plane.

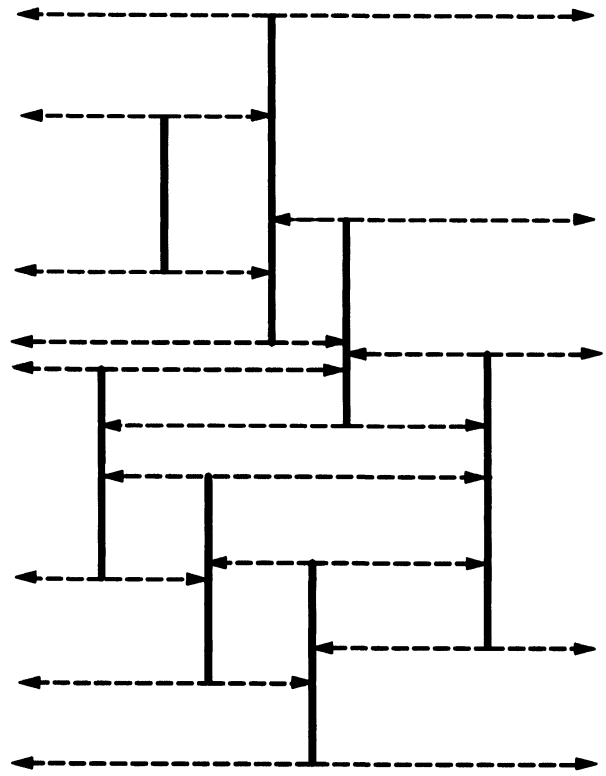

FIGURE 7 The trapezoidal decomposition of line segments.

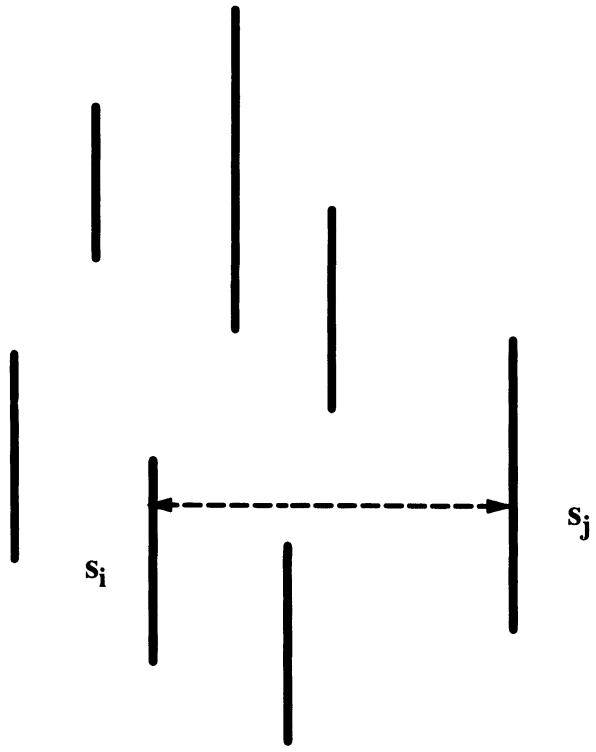

FIGURE 6 A collection of vertical line segments; segments s_i and s_j see each other.

Lemma 4.1. Let s_i and s_j be arbitrary vertical segments. The segments s_i and s_j see each other if, and only if, there exists a trapezoid T whose vertical edges are subsegments of s_i and s_j.

Proof. To begin, note that if s_i and s_j see each other, then the existence of a trapezoid satisfying the requirements of the lemma follows directly from the definition. Conversely, suppose that there exists a trapezoid T satisfying the conditions of the lemma. It is easy to see that no other vertical line segment can be between s_i and s_j, and the conclusion follows. □

Lemma 4.1 motivates the following approach to constructing the set of all visible pairs: perform the trapezoidal decomposition of the plane and then assign one processor to each corner of every trapezoid in the decomposition. Every processor is responsible for detecting whether the trapezoid it has been assigned to is *bounded* i.e. satisfies the requirements of Lemma 4.1. The visibility graph will be returned by its adjacency lists. The details are spelled out as follows.

Algorithm Construct_Visibility_Graph;
{Input: a collection $S = \{s_1, s_2, \ldots, s_n\}$ of n vertical line segments in the plane;
Output: the corresponding visibility graph represented by adjacency lists;}

Step 1. Sort all the vertical line segments in increasing order of their x-coordinates;

Step 2. Perform a trapezoidal decomposition of the plane by using the algorithm of [1];

Step 3. Assign one processor to each corner of every trapezoid formed in this way; for $(1 \leq i \leq n)$ let T_i be such a trapezoid and let $P_1^i, P_2^i, P_3^i, P_4^i$ be the processors assigned to T_i, with P_j^i assigned to the upper-left corner and proceeding counter-clockwise (see Figure 8);

Step 4. For all i, j $(1 \leq i \leq n;\ 1 \leq j \leq 4)$ in parallel processor P_j^i identifies the following trapezoidal line segments:

- a_j^i starting from the point on which P_j^i is assigned and pointing right (left);
- c_j^i the nearest trapezoidal line below a_j^i and parallel to a_j^i;
- b_j^i and d_j^i the two vertical line segments which bound a_j^i and c_j^i (see Figure 8 for an illustration);

Step 5. For all i, j $(1 \leq i \leq n;\ 1 \leq \mathrm{j} \leq 4)$ processor P_j^i marks itself *active* if the rectangular region determined by a_j^i, b_j^i, c_j^i, and d_j^i is bounded (i.e. b_j^i and d_j^i both exist); otherwise, P_j^i marks itself *inactive* and will never be active again;

Step 6. Every active processor P_j^i records (p, q) in its own memory, with $s_p = b_j^i$ and $s_q = d_j^i$;

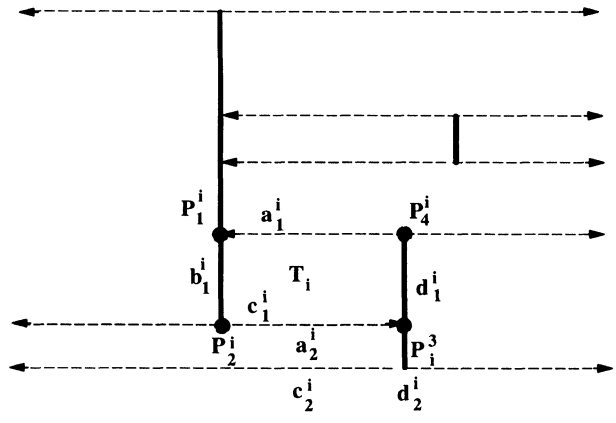

FIGURE 8 Illustrating Steps 3 and 4 of Construct_Visibility_Graph.

Step 7. Sort the ordered pairs (p, q) produced in Step 6 by their first component.

Step 8. In the sorted list mark the first element in every group of ordered pairs having the same first component; what results are the adjacency lists of the visibility graph.

To summarize our findings we state the following result.

Theorem 4.2. The visibility graph of a collection of n disjoint vertical line segments in the plane can be constructed in O(log n) time using O(n) processors in the CREW model of computation.

Proof. The correctness follows directly from Lemma 4.1. Therefore, we turn our attention to the complexity. First, we note that since no write conflicts arise, the computation can be performed in the CREW model.

To begin, we note that the number of trapezoids in the decomposition is linear in the number of segments [11]. Step 1 takes O(log n) time using O(n) processors if we use Cole's sorting algorithm. Next, Step 2 uses the algorithm in [1] running in O(log n) time and using O(n) processors in the CREW model. Step 3 only involves assignments and takes O(1) time. Note that in Step 4 every processor P_j^i takes O(1) time to determine a_j^i, b_j^i, c_j^i, and d_j^i since the required information is stored in one trapezoid in the decomposition; once this information is known, P_j^i marks itself active only if the corresponding trapezoid is bounded: this is done in O(1) time; finally, Steps 7 and 8 can be implemented in O(log n) time with O(n) processors by using Cole's sorting algorithm. □

Theorem 4.2, Proposition 2.3, and Theorem 2.6 combined imply the following result.

Theorem 4.3. Consider a collection of n iso-oriented, non-overlapping, rectangles $R_1, R_2, \ldots, R_n$ in the plane. The corresponding visibility graph can be constructed in O (log n) time using O (n) processors in the CREW. Furthermore, this is both time- and cost-optimal in this model of computation.

5 CONCLUDING REMARKS

In this paper we presented time- and cost-optimal parallel algorithms to construct two special kinds of graphs motivated by, and finding applications to, the compaction step of integrated circuit design. Specifically, we have shown that with a family of n iso-oriented, non-overlapping rectangles in the plane, both the dominance and the visibility graphs can be constructed in O(log n) time with O(n) processors in the CREW model of computation.

We have also shown that these results are best possible both with respect to the amount of work and with respect to the running time. In fact, we have shown that the running time of these algorithms cannot be further reduced even if an infinite number of processors and memory cells are available.

Both our algorithms are conceptually very simple, relying on the trapezoidal decomposition developed in [1]. Our algorithms are, in fact, new evidence to the elegance and power of the trapezoidal decomposition.

We close with the obvious question: what other compaction-related problems can be solved by using this decomposition? This is a promising area for further research.

Acknowledgement

The authors would like to express their appreciation to two anonymous referees for their very careful reading of the first version of the manuscript and for numerous constructive comments.

References

[1] M. J. Atallah, R. Cole and M. T. Goodrich, Cascading divide-and-conquer: a technique for designing parallel algorithms, *SIAM Journal on Computing*, (18), 1989 499–532.

[2] R. Cole, Parallel Merge Sort, *SIAM Journal on Computing*, 17 (1988) 770–785.

[3] S. A. Cook, C. Dwork, and R. Reischuk, Upper and lower time bounds for parallel random access machines without simultaneous writes, *SIAM Journal on Computing*, 15 (1986) 87–97.

[4] D. Dobkin and R. Lipton, On the complexity of computations under varying set of primitives, *Journal of Computer and Information Sciences*, 18 (1979), 86–91.

[5] E. Lodi and L. Pagli, A VLSI solution to the Vertical Segment Visibility Problem, *IEEE Transactions on Computers*, C-35, (1986), 923–928.

[6] F. Luccio, S. Mazzone, and C. K. Wong, A note on visibility graphs, *Discrete Mathematics*, 64, (1987), 209–219.

[7] J. JáJá, *An introduction to parallel algorithms*, Addison-Wesley, Reading, MA, 1991.

[8] A. A. Malik, An efficient algorithm for generation of constraint graph for compaction, *Proc. International Conference on CAD*, 1987, 130–133.

[9] C. A. Mead and L. Conway, *Introduction to VLSI Systems*, Addison-Wesley, Reading MA, 1979.

[10] B. T. Preas and M. J. Lorenzetti (Eds.) *Physical Design Automation of VLSI Systems*, Benjamin/Cummings, Menlo Park, 1988.

[11] F. P. Preparata and M. I. Shamos, *Computational Geometry : An Introduction*, Springer-Verlag, 1985.

[12] M. Schlag, F. Luccio, P. Maestrini, D. T. Lee, and C. K. Wong, A visibility problem in VLSI layout compaction, in Franco P. Preparata, Ed., *Advances in Computing Research*, Vol. 2, (1984), 259–282.

[13] J. D. Ullman, *Computational Aspects of VLSI*, Computer Science Press, 1984.

[14] W. H. Wolf and A. E. Dunlop, Symbolic layout and compaction, in [10], 211–281.

Biographies

DHARMAVANI BHAGAVATHI Dr. Bhagavathi obtained the B.E. degree in Electrical Engineering in 1987 from Osmania University, the M.Tech. in Computer Science from University of Hyderabad, in 1989, and the Ph.D. degree in Computer Science from Old Dominion University in 1993.
Since Fall 1993 Dr. Bhagavathi is an Assistant Professor with the Department of Computer Science of Southern Illinois University at Edwardsville. Her current research interests include image processing, computational geometry, and parallel architectures.

HIMABINDU GURLA Himabindu Gurla received her Bachelor of Engineering degree in Computer Science from Osmania University College of Engineering, India, in 1991, and is currently a Ph.D. candidate at Old Dominion University. Her research interests include parallel algorithms, parallel architectures, systems programming and compilers.

STEPHAN OLARIU Dr. Olariu received the M.Sc. and Ph.D. degrees in computer science from McGill University, Montreal in 1983 and 1986, respectively. In 1986 he joined the Computer Science Department at Old Dominion University where he is now an Associate Professor.
Dr. Olariu has published more than 120 papers in various journals and conference proceedings. His research interests include image processing and machine vision, parallel architectures, design and analysis of parallel algorithms, computational graph theory, computational geometry, and computational morphology.

JAMES SCHWING James Schwing (M '83) received a B.S. degree in mathematics from Worcester Polytechnic Institute in 1970 and a Ph.D in mathematics from the University of Utah in 1976. In 1976, he joined the faculty of Old Dominion University in the Department of Computer Science where he is now a Professor of Computer Science. Since 1983, he has been Assistant Chair. His research ionterests include parallel algorithms, computer graphics, computer aided design and human-computer interaction. Dr. Schwing is a member of the Association of Computing Machines.

JINGYUAN ZHANG Jingyuan Zhang received his BS degree in computer Science from Shandong University in 1984, his MS degree in computer Science fron Zhejiang University in 1987 and his Ph.D from Old Dominion University in 1992. From June 1987 to August 1989, he was an instructor with the Department of Electrical Engineering and computer Science at Ningbo University. Currently he is an Assistant Professor with the Department of Mathematics and Computer Science, Elizabeth City State University. His research interests include parallel algorithms, reconfigurable architectures, computer graphics, and computer aided design.